TP6 : Algorithme de Knuth-Morris-Pratt et bords maximaux

Objectifs du TP :

- Implémenter et analyser une version efficace de l'algorithme de Knuth-Morris-Pratt.
- Réviser la manipulation de chaînes de caractères en C.
- Exploiter la notion de bord maximal pour résoudre d'autres problèmes d'algorithmique du texte.

L'objectif premier de ce TP est de résoudre le problème suivant : étant donné un texte t et un mot m, déterminer où se trouvent les occurrences de m dans t. Par exemple, si $t = ababaaaba$ et que $m = aba$, on souhaite écrire un algorithme qui affiche successivement :

Le motif apparaît en position 0
Le motif apparaît en position 2
Le motif apparaît en position 6

Vous connaissez déjà plusieurs algorithmes permettant de répondre à ce problème de recherche de motif dans un texte. L'algorithme de Knuth-Morris-Pratt en est un : il repose sur la construction de l'automate des occurrences associé au motif, dans lequel il suffit de lire t pour détecter la présence de m dans t. Ce TP vise à implémenter une version efficace de l'algorithme de Knuth-Morris-Pratt dans laquelle l'automate des occurrences n'apparaît pas explicitement.

Les questions 1 à 8c incluses sont à traiter en C et à rendre au format papier pour le 18/11. Dans tout le sujet, on pourra utiliser les fonctions classiques du module `string`, notamment `strlen` et `strcmp`. On considèrera que m et t sont écrits sur un même alphabet Σ constitué des caractères ASCII privés du symbole $\#$. Si $u \in \Sigma^\star$ est un mot de longueur k on numérotera les lettres de u à partir de 0 et ainsi $u = u_0 ... u_{k-1}$.

Partie 1 *Calcul efficace de bords maximaux*

On appelle *bord* d'un mot $u \in \Sigma^+$ tout mot différent de u et qui est à la fois préfixe et suffixe de u. Le *bord maximal* d'un tel mot est l'unique bord de u de longueur maximale et on le note $B(u)$. Par convention, on considère que $B(\varepsilon) = \varepsilon$.

1. Indiquer quels sont les bords de $u = abaababa$ et calculer $B(u)$.

On cherche dans la suite de cette partie à répondre au problème suivant : étant donné un mot u, déterminer pour tout $i \in [\![0, |u|]\!]$ le bord maximal du préfixe de taille i de u. On constate que pour ce faire, il suffit de déterminer la longueur du bord maximal de chacun des préfixes de u ; en effet, le bord maximal d'un mot m est un préfixe de m donc connaître sa longueur et m suffit à connaître $B(m)$.

2. Ecrire une fonction `char* sous_chaine(char* m, int i, int j)` renvoyant le facteur $m_i ... m_j$ du mot `m` compris entre les positions `i` et `j` incluses (on supposera sans le vérifier que `i` et `j` sont licites).
3. Ecrire une fonction naïve `int bord_maximal(char* m, int i)` renvoyant la longueur du bord maximal du préfixe de taille `i` du mot `m`.
4. En déduire une fonction `int* bords_maximaux(char* m)` calculant un tableau dont la case i contient la taille du bord maximal du préfixe de taille i du mot `m`. Par exemple, `bords_maximaux("abaababa")` devrait être le tableau `[0,0,0,1,1,2,3,2,3]`. Quelle est la complexité temporelle de cette fonction ?

Pour améliorer cette complexité on utilise un algorithme dynamique pour le calcul des longueurs des bords maximaux des préfixes de u. Si $u \neq \varepsilon$, on note l_i la longueur du bord maximal du préfixe de taille i de u, ce pour tout $i \in [\![0, |u|]\!]$.

5. On suppose dans ces questions que $u \neq \varepsilon$.
   a) Montrer qu'il existe $k \leq |u|$ tel que l'ensemble des bords de u soit $\{B(u), B(B(u)), B^3(u), ..., B^k(u)\}$.
   b) Montrer que si u est de longueur k et a est une lettre alors $B(ua)$ est le plus long des mots de $\{B(u)a, B^2(u)a, ..., B^k(u)a, \varepsilon\}$ qui sont préfixes de u.

La question précédente explique comment calculer l_i si on connaît déjà les valeurs de l_j pour $j < i$. En effet, le préfixe de taille i de u est $p = va$ avec v le préfixe de taille $i-1$ de u. La question 5b nous dit que $B(p) = B(va)$ vaut soit ε soit est de la forme wa avec w un bord de v. Puisqu'on veut le plus grand, on commence par regarder si $B(v)a$ est préfixe de p (il en est immédiatement suffixe) : pour ce faire, il suffit de savoir si a est égal à la lettre en position l_{i-1}. Si on a égalité, $l_i = l_{i-1} + 1$, sinon, on réitère ce test avec le bord suivant de v, à savoir $B^2(v)$ dont la taille est un l_j connu puisque $B(v)$ est un préfixe de v donc un préfixe de u de taille strictement inférieure à i ! On obtient ainsi pour le calcul des longueurs des bords maximaux le pseudo-code suivant :

1\. bords_maximaux(m) =
2\.   Initialiser un tableau L de taille $|m|+1$ rempli de 0
3\.   Pour i allant de 2 à $|m|$
4\. $l \leftarrow L[i-1]$
5\.   Tant que $l > 0$ et $m_l \neq m_{i-1}$
6\.   *//la dernière lettre du préfixe de taille i est* m_{i-1}
7\. $l \leftarrow L[l]$
8\.   Si $m_l = m_{i-1}$ *//Cas où* $B(m_0...m_{i-1})$ *est un des* $B^r(m_0...m_{i-2})a$
9\. $L[i] \leftarrow l+1$
10\.   Sinon //Cas où $B(m_0...m_{i-1}) = \varepsilon$
11\. $L[i] \leftarrow 0$
12\.   Renvoyer L

6. Ecrire une fonction `int* bords_maximaux_dyn(char* m)` ayant la même spécification que `bords_maximaux` mais utilisant la stratégie précédente.

7. Expliquer pourquoi le nombre **total** de passages dans la boucle tant que de la ligne 5 lors d'un appel à `bords_maximaux_dyn` sur un mot m est majoré par $|m|$. En déduire la complexité de `bords_maximaux_dyn`.

Partie 2 *Algorithmique du texte autour des bords maximaux*

8. On considère le mot $m\#t$ de $\Sigma \cup \{\#\}$ (rappelons que $\#$ ne fait pas partie de l'alphabet Σ).
   a) Expliquer comment le calcul des bords maximaux de tous les préfixes de $m\#t$ permet de détecter toutes les occurrences du motif m dans le texte t. On ne demande pas une preuve rigoureuse.
   b) En déduire une fonction `void KMP(char* m, char* t)` produisant l'affichage de toutes les positions des occurrences du motif `m` dans le texte `t` selon les modalités de l'introduction.
   c) Quelle est la complexité temporelle de cette fonction ? Combien d'espace demande-t-elle en plus de celui occupé par l'entrée ? Comparer cette quantité avec celle que nécessiterait le stockage de l'automate des occurrences. Commenter.
   d) (bonus) Où se cache l'automate des occurrences dans la fonction `KMP` ?

Remarquez que la fonction `KMP` peut facilement être adaptée pour compter les occurrences du motif m, renvoyer la première ou la dernière position de ce motif, ou simplement renvoyer un booléen indiquant si le motif m est présent dans t, ce sans modifier les coûts de la question 8c. Les questions suivantes présentent quelques autres applications au calcul des bords maximaux.

9. Deux mots u et v sont dits conjugués s'il existe deux mots r et s tels que $u = rs$ et $v = sr$. Ecrire une fonction `bool sont_conjugues(char* u, char* v)` qui détermine en temps $O(|u| + |v|)$ si deux mots sont conjugués. On expliquera la correction de cette fonction et on justifiera sa complexité.

10. Ecrire une fonction `bool carre(char* u)` indiquant si le mot `u` contient un facteur carré (non vide). On garantira pour cette fonction une complexité en $O(|u|^2)$ qu'on justifiera en même temps que la correction de la fonction demandée. *Indication : Examiner les bords des conjugués de u.*

11. Une période d'un mot u est un préfixe $v \neq \varepsilon$ de u tel que u soit préfixe de v^n pour un certain $n \geq 1$. Par exemple, ab est une période de $abababa$.
    a) Si $u = vw$ avec $v \neq \varepsilon$, montrer que v est une période de u si et seulement si w est un bord de u.
    b) Ecrire une fonction `char* periode(char* u)` qui détermine la plus petite période de `u` en temps linéaire.

12. En exploitant la notion de bord, expliquer comment déterminer la liste de tous les préfixes d'un mot qui sont des palindromes.